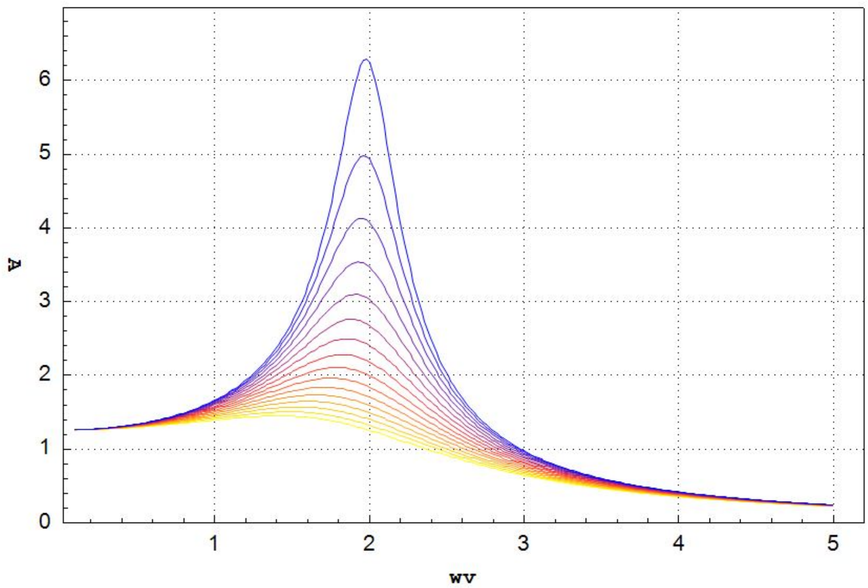
Pendolo a Torsione

Francesco Forcher
Università di Padova, Facoltà di Fisica
francesco.forcher@studenti.unipd.it
Matricola: 1073458

Davide Chiappara
Università di Padova, Facoltà di Fisica
davide.chiappara@studenti.unipd.it
Matricola: 1073458

Simone Frau
Università di Padova, Facoltà di Fisica
simone.forcher@studenti.unipd.it
Matricola: 1073458

30 ottobre 2014



Sommario

Obiettivo dell'esperienza è stato studiare il moto di un cilindro in acqua in risposta ad una forza periodica impressa al clindro stesso da un motore elettrico attraverso un filo che permette la trasmissione di un momento torcente. Dai campioni ricavati sperimentalmente si è indagato, in particolare, sulla curva di risonanza, sulla frequenza di risonanza, sul coefficiente di smorzamento e sulla pulsazione propria del sistema.

Indice

I. Apparato strumentale

L'apparato strumentale consiste in un cilindro in plexiglass al cui interno è posto un peso in acciaio di massa: $(115.5 \pm 0.1)\mathtt{g}$ e diametro: $(22.7 \pm 0.1)\mathtt{mm}$ e altezza: $(34.0 \pm 0.1)\mathtt{mm}$ collegato ad un filo di acciaio armonico, materiale dotato discrete capacità elastiche, ed immerso in acqua. Il filo è inoltre collegato ad una piattaforma rotante azionata da un motore di diametro circa 8 cm che, una volta azionato, induce un'oscillazione sul corpo formato dal filo più il pesetto. Il range delle frequenze alla quale il motore può essere indotto ad oscillare risulta compreso tra 0.800 e 1.200 Hz. Di suddetta oscillazione è possibile modificare il periodo e l'ampiezza può essere impostata da 2 millesimi di giro a 16.

Il tutto viene controllato e registrato mediante l'interfaccia fornita da un computer. I dati vengono acquisiti in intervalli di 0.05 secondi permettendo una frequenza di rilevamento di 20 dati per secondo. L'interfaccia permette di visualizzare valori della frequenza, dell'ampiezza. Inoltre sono presenti diversi grafici, il più intereante dei quali rappresenta l'angolo di cui è ruotao il pendolo in funzione del tempo. Infine la presenza del pulsante offset permette di tarare l'apparato dopo ogni misurazione, al fine di limitare errori sistematici.

II. Metodologia di misura

Per poter stimare la frequenza di risonanza si è proceduto azionando il motore e mettendo in oscillazione la piattaforma rotante. Partendo dalle informazioni fornite e dall'apparecchiatura si è deciso di porre l'ampiezza a 10 millesimi di giro e di variare gli intervalli delle frequenze di 0.020 Hz. Sono stati acquisiti campioni di misure per un tempo totale di circa 10 secondi durante la fase a regime, è stata interotta la misurazione per eseguire lo store dei dati così ricavati, è stato spento il motore e si è intrapresa una seconda fase di registrazione dati per una durata di circa 20 secondi per la fase di smorzamento, al termine del quale è stato eseguito nuovamente lo store dei dati. Le misure sono state prese non appena è stato evidente dai grafici di riferimento il carattere periodico del moto del pendolo.

L'apparato strumentale è stato ricalibrato prima delle prese dati della giornata per ricavarne un funzionamento ottimale. L'ampiezza massima di oscillazione della forzante è stata scelta per permettere oscillazioni abbastanza ampie da studiare ma non così ampie da rendere caotico il moto del pendolo, costringendolo a muoversi sul piano perpendicolare all'asse di oscillazione. Attraverso questo metodo è stato possibile ottenere una panoramica del comportamento oscillatorio del corpo di studio e identificare efficacemente il settore in cui avveniva il fenomeno di risonanza. Tale settore è stato poi sondato ricorrendo al metodo di bisezione restringendosi in un intorno di valori della frequenza e aumentando l'esposizione dell'acquisizione dati.

In questa fase (che si concentra nell'intervallo tra 0.965 Hz e 0.970 Hz) sono stati registrati valori per la durata di circa 100 secondi per la fase a regime e di 40 secondi circa per la fase in smorzamento, una volta spento il motore. Questo ha offerto agli sperimentatori la possibilità di analizzare una serie di campioni più concentrati avente intervalli di frequenza di 0.001 Hz, permettendo di stimare il più efficacemente possibile la frequenza di risonanza.

III. Presentazione dei dati

I. Tabelle

Tabella 1: *Pulsazioni smorzate*

Frequenza forzante [Hz]	Pulsazione smorzante $\left[\frac{\text{rad}}{\text{s}}\right]$	Errore $\left[\frac{\text{rad}}{\text{s}}\right]$
0.900	6	1
0.920	6.1	0.2
0.940	6.1	0.2
0.960	6.08	0.03
0.965	6.1	0.1
0.966	6.07	0.06
0.967	6.08	0.06
0.968	6	1
0.969	5	2
0.970	6.08	0.08
0.975	6.0	0.6
0.980	6.08	0.06
0.990	6.1	0.1
1.000	6	1
1.020	6.1	0.5
1.060	5	2
1.080	6	1

Media pesata pulsazione di smorzamento:$(6.07 \pm 0.09)[\frac{rad}{s}]$

Tabella 2: *Ampiezze di oscillazione a regime*

Frequenza forzante [Hz]	Ampiezza [giri]	σ_{amp} [giri]
0.900	0.031	0.001
0.920	0.0452	0.0009
0.940	0.083	0.001
0.960	0.2187	0.0007
0.965	0.247	0.008
0.966	0.23	0.01
0.967	0.282	0.002
0.968	0.290	0.006
0.969	0.290	0.002
0.970	0.275	0.002
0.975	0.227	0.001
0.980	0.168	0.003
1.000	0.072	0.003
1.020	0.050	0.002
1.040	0.0373	0.0007
1.060	0.0302	0.0005
1.080	0.0257	0.0007

Tabella 3: *Interpolazione per trovare le gamma, retta* $y = a + b \cdot x$ *su scala logaritmica*

Frequenza forzante [Hz]	Parametro a	Errore su a	Parametro b [Hz]	Errore su b [Hz]
0.900	-1.56	0.02	-0.045	0.002
0.920	-1.22	0.04	-0.047	0.003
0.940	-0.82	0.05	-0.040	0.005
0.960	0.293	0.009	-0.0482	0.0008
0.965	0.459	0.006	-0.0481	0.0003
0.966	0.461	0.005	-0.0481	0.0002
0.967	0.51	0.02	-0.0470	0.0008
0.968	0.53	0.02	-0.0467	0.0004
0.969	0.52	0.03	-0.0469	0.0005
0.970	0.42	0.04	-0.045	0.002
0.975	0.23	0.02	-0.046	0.001
0.980	-0.14	0.06	-0.039	0.005
1.000	-0.85	0.09	-0.035	0.008
1.020	-1.25	0.02	-0.045	0.002
1.060	-2.05	0.02	-0.048	0.002
1.080	-2.2	0.1	-0.03	0.01

Media pesata gamma: $(-0.047 \pm 0.007)[\mathsf{Hz}]$
Pulsazione propria: $(6.1 \pm 0.3)[\frac{rad}{s}]$

II. Grafici

IV. Analisi dei dati

Come detto nella descrizione dell'apparato strumentale, il tasso di rilevamento dei dati è di 20 al secondo. Questo corrisponde a una frequenza di campionamento di 20 Hertz, di molto superiore al Nyquist rate necessario per il pendolo (il doppio della massima frequenza campionabile), dato che come verificabile a vista ha una frequenza dell'ordine di 1 Hz. Non ci sono quindi problemi di aliasing e sottocampionamento. Per quanto riguarda l'offset, è stata rifatta la calibrazione prima di ogni presa dati (inizio giornata) e si può vedere che lo strumento era calibrato da un'evidente simmetria rispetto all'asse delle ascisse. Per il calcolo dei massimi è stato utilizzato un programma che riconoscesse i punti di massimo e minimo approssimando la funzione come una parabola in un intorno dei dati stazionari (dati massimi e minimi locali) usando il dato precedente e il successivo, vincolando la parabola a passare per questi 3 punti e trovandone il vertice. L'errore legato all'utilizzo di questa approssimazione è, come noto dallo sviluppo di Taylor delle funzioni goniometriche, $o(x^3)$ che, essendo lo step 0.05 s è dell'ordine di 10^{-3}, e trascurabile rispetto agli altri errori. Per una stima delle ampiezze legate alle frequenze di oscillazione sono stati presi i valori medi delle ordinate dei massimi (e dei valori assoluti dei minimi).

Una stima della pulsazione di risonanza è stata fatta con un processo di esplorazione iniziale che ha permesso, attraverso il metodo di bisezione, di concentrarsi sull'area nella quale l'ampiezza era più alta. Il valore finale trovato risulta di $(0.968 \pm 0.001)[\mathrm{Hz}]$. Tale valore è stato scelto in quanto valore per il quale l'ampiezza misurata risultava più alta. L'errore preso è la precisione dello strumento. Sono state abbandonate idee differenti sulla stima di questo valore in quanto dato l'apparato sperimentale l'errore non può essere ridotto. Per stimare il coefficiente di smorzamento γ legato al forza viscosa dell'acqua è stato tentato un approccio diretto con gnuplot, ma i problemi del suo algoritmo (Levenberg–Marquardt, una forma di step gradient descent) nel caso di funzioni come questa, in cui il gradiente dei minimi quadrati è pieno di punti stazionari locali, ne hanno impedito l'applicabilità pratica. Quindi è stato scelto un altro approccio che elimini questi problemi, in particolare limitando lo studio a una semplice funzione esponenziale, che è stata ulteriormente semplificata in un fit lineare usando una scala logaritmica. Di consequenza, sono stati cercati gli $x_i \mid f(x_i) = 0$. Essendo la funzione che descrive l'angolo in assenza della forzante

$$\theta(t) = \theta_0 e^{-\gamma t} \sin(\omega_s t + \phi), \tag{1}$$

poichè $e^{-\gamma t} > 0$ gli zeri della funzione sono solo gli zeri del seno. Quindi i punti medi $x_m = \frac{x_i + x_{i+1}}{2}$ fra gli zeri sono i punti in cui $sin(\omega_s t + \phi) = 1$. Interpolando questi punti la funzione diventa dunque

$$\theta_0 e^{-\gamma t} \cdot 1 = \theta_0 e^{-\gamma t}.$$

Interpolando questa funzione con i punti $(x_m, \log f(x_m))$ (le coordinate $f(x_m)$ sono state calcolate, nei casi in cui non fossero già un punto dei dati, approssimando linearmente tra i due punti più vicini).

La pulsazione di smorzamento è stata ottenuta attraverso una media pesata delle pulsazioni ottenute dallo studio dei periodi dei grafici durante la fase di smorzamento (vedasi **Tabella 1**). Gli errori sono stati stimati a partire da una stima diretta con la sommatoria degli scarti al quadrato diviso $N-1$. La pulsazione propria è stata trovata attraverso la formula $\omega_0 = \sqrt{{\omega_s}^2 + \gamma^2}$ e il suo errore è stato trovato per propagazione. I grafici rivelano che, entro gli errori casuali, l'analisi dati compiuta risulta in linea con ciò che ci si aspettava. Asperità presenti nel grafico possono essere connesse o a spike momentanei del sistema di misurazione o a movimenti bruschi che ne hanno alterato il perfetto funzionamento. Dalla curva di risonanza, in particolare, si riconosce che i dati che peggio approssimano una distribuzione teorica sono quelli che presentano un errore più elevato. Sebbene alcuni risultati dell'analisi dati non sembrino rispecchiare al meglio la curva sperimentale non si riconosce in essi un errore sistematico che possa portare a un errata stima dei risultati ottenuti, ad esempio nei grafici del moto a regime tutti i massimi sembrano leggermente spostati a destra ma ciò non influenza né il calcolo della frequenza né quelo dell'ampiezza. L'apparato strumentale, comunque, permetteva un'ottima ricerca attorno al valore di risonanza, infatti la maggior parte dei dati non perfettamente coerenti sono a basse o alte frequenze, o comunque lontani dalla frequenza di risonanza.

V. Conclusioni

L'esperimento ha creato dei risultati che bene si accordano con le previsioni sperimentali (per esempio, si può vedere dal fatto che i coefficienti di smorzamento sono molto simili per tutte le prove effettuate). La curva di risonanza si rivela un buono strumento per lo studio delle ampiezze in funzione della frequenza, e il grafico da essa disegnato non si discosta molto da quello atteso.

Per migliorare I risultati ottenuti, sarebbe stato necessario ridurre le interazioni dell'ambiente con il pendolo (impossibile con l'apparato strumentale dato) oppure effettuare un maggior numero di indagini anche a frequenze differenti. L'analisi dati è stata fatta in modo da minimizzare gli errori, che risultano comunque molto buoni per tutti i risultati presentati.

VI. Codice

```
/*
 * approx_lineare.cpp

 *
 *  Created on: Jun 9, 2014
 *      Author: francesco
 */

#include <vector>
using std::vector;

class funzione_punti_lineare {
```

```
  std::vector<long double> vectx;
  std::vector<long double> vecty;

  //x dev'essere ordinato?
  funzione_punti_lineare(std::vector<long double> vx, std::vector<long double> vy) {
    vectx = vx;
    vecty = vy;
  }

  long double operator()(double x) {
    long long i = 0;
    while(vectx.at(i) <= x)
      i++;

    long double x0 = vectx.at(i);
    long double y0 = vecty.at(i);

    long double x1 = vectx.at(i+1);
    long double y1 = vecty.at(i+1);

    //Coeff. angolare, DeltaY/DeltaX
    long double m = (y1 - y0) / (x1 - x0);
    long double y = m*(x-x0)+y0;

    return y;
  }
};
```

../src/approx_lineare.cpp

```
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;
int main()
{
  int n = 0;
  vector <double> valori;
  double temp;             //Variabile temporanea contenente i valori
  double temp_neg;           //Variabile temporanea ausiliaria 1
  double temp_pos;           //Variabile temporanea ausiliaria 2
  while(cin >> temp)           //Mette i dati nel vector
  {
    valori.push_back(temp);
    n++;
  }
  int j = 0;
  vector <double> massimi;    //Crea vector in cui mettere i massimi
  vector <double> max_neg;    //Crea vector in cui mettere elemento precedente al massimo
  vector <double> max_pos;    //Crea vector in cui mettere elemento successivo al massimo
```

```
vector <int> max_dist;       //Crea vector in cui scrivere posizione massimi
vector <int> max_dist_neg;
vector <int> max_dist_pos;
double temp_dist;       //Crea variabile temporanea per la distanza dei valori
double temp_dist_neg;
double temp_dist_pos;

for (int i = 1 ; i < ( n - 1 ) ; i++)
{
  if (valori.at(i) > valori.at(i-1) && valori.at(i) >= valori.at(i+1) ) //Trova i massimi
  {
    temp = valori.at(i);
    temp_neg = valori.at(i-1);
    temp_pos = valori.at(i+1);
    temp_dist = (i + 1);
    temp_dist_neg = i ;
    temp_dist_pos = i + 2;
    massimi.push_back(temp);
    max_neg.push_back(temp_neg);
    max_pos.push_back(temp_pos);
    max_dist.push_back(temp_dist);
    max_dist_neg.push_back(temp_dist_neg);
    max_dist_pos.push_back(temp_dist_pos);

    j++;

  }
}
vector <double> minimi;
vector <double> min_neg;
vector <double> min_pos;
vector <int> min_dist;
vector <int> min_dist_neg;
vector <int> min_dist_pos;
j = 0;
for (int i = 1 ; i < ( n - 1 ) ; i++)
{
  if (valori.at(i) < valori.at(i-1) && valori.at(i) <= valori.at(i+1) )
  {
    temp = valori.at(i);
    temp_neg = valori.at(i-1);
    temp_pos = valori.at(i+1);
    temp_dist = i + 1;
    temp_dist_neg = i ;
    temp_dist_pos = i + 2;
    minimi.push_back(temp);
    min_neg.push_back(temp_neg);
    min_pos.push_back(temp_pos);
    min_dist.push_back(temp_dist);
    min_dist_neg.push_back(temp_dist_neg);
```

```
      min_dist_pos.push_back(temp_dist_pos);

      j++;
    }
  }



  double delta, da, db, dc, a, b, c, vortex, vortey;     //Parametri della parabola
  vector <double> xverticiMAX;            //Vector contenente i risultati del programma
  vector <double> yverticiMAX;

  double max_size = massimi.size();
  double min_size = minimi.size();

  for (int i = 0 ; i < max_size ; i++)
  {
    delta = ( ( max_dist.at(i)      * max_dist_pos.at(i) * max_dist_pos.at(i) ) +
            ( max_dist_neg.at(i) * max_dist.at(i)      * max_dist.at(i) ) +
            ( max_dist_neg.at(i) * max_dist_neg.at(i) * max_dist_pos.at(i) ) -
            ( max_dist_neg.at(i) * max_dist_neg.at(i) * max_dist.at(i) ) -
            ( max_dist.at(i)      * max_dist.at(i)      * max_dist_pos.at(i) ) -
            ( max_dist_neg.at(i) * max_dist_pos.at(i) * max_dist_pos.at(i) ) );


  da = (  ( max_neg.at(i) * max_dist.at(i)      * max_dist_pos.at(i) * max_dist_pos.at(i) ) +
      ( max_pos.at(i) * max_dist_neg.at(i) * max_dist.at(i)      * max_dist.at(i) )      +
      ( massimi.at(i) * max_dist_neg.at(i) * max_dist_neg.at(i) * max_dist_pos.at(i) ) -
      ( max_pos.at(i) * max_dist_neg.at(i) * max_dist_neg.at(i) * max_dist.at(i) )      -
      ( max_neg.at(i) * max_dist.at(i)      * max_dist.at(i)      * max_dist_pos.at(i) ) -
      ( massimi.at(i) * max_dist_neg.at(i) * max_dist_pos.at(i) * max_dist_pos.at(i) ) );


  db = (  ( massimi.at(i) * max_dist_pos.at(i) * max_dist_pos.at(i) ) +
      ( max_neg.at(i) * max_dist.at(i)      * max_dist.at(i) )      +
      ( max_pos.at(i) * max_dist_neg.at(i) * max_dist_neg.at(i) ) -
      ( massimi.at(i) * max_dist_neg.at(i) * max_dist_neg.at(i) ) -
      ( max_pos.at(i) * max_dist.at(i)      * max_dist.at(i) )      -
      ( max_neg.at(i) * max_dist_pos.at(i) * max_dist_pos.at(i) )    );
```

```


  dc = (  ( max_pos.at(i) * max_dist.at(i) )      +
      ( massimi.at(i) * max_dist_neg.at(i) ) +
      ( max_neg.at(i) * max_dist_pos.at(i) ) -
      ( max_neg.at(i) * max_dist.at(i) )      -
      ( massimi.at(i) * max_dist_pos.at(i) ) -
      ( max_pos.at(i) * max_dist_neg.at(i) )     );

    a = da / delta;
    b = db / delta;
    c = dc / delta;
    vortex = - b / ( 2 * c );
    vortey = - ( b * b - 4 * a * c ) / ( 4 * c );
    xverticiMAX.push_back(vortex * 0.05); //Presente valore di conversione delle x
    yverticiMAX.push_back(vortey);
  }


  vector <double> xverticiMIN;
  vector <double> yverticiMIN;

  for (int i = 0 ; i < min_size ; i++)
  {
    delta = ( ( min_dist.at(i)      * min_dist_pos.at(i) * min_dist_pos.at(i) ) +
            ( min_dist_neg.at(i) * min_dist.at(i)      * min_dist.at(i) )     +
            ( min_dist_neg.at(i) * min_dist_neg.at(i) * min_dist_pos.at(i) ) -
            ( min_dist_neg.at(i) * min_dist_neg.at(i) * min_dist.at(i) )     -
            ( min_dist.at(i)      * min_dist.at(i)      * min_dist_pos.at(i) ) -
            ( min_dist_neg.at(i) * min_dist_pos.at(i) * min_dist_pos.at(i) ) );


  da = (  ( min_neg.at(i) * min_dist.at(i)      * min_dist_pos.at(i) * min_dist_pos.at(i) ) +
      ( min_pos.at(i) * min_dist_neg.at(i) * min_dist.at(i)      * min_dist.at(i) )      +
      ( minimi.at(i)  * min_dist_neg.at(i) * min_dist_neg.at(i) * min_dist_pos.at(i) ) -
      ( min_pos.at(i) * min_dist_neg.at(i) * min_dist_neg.at(i) * min_dist.at(i) )      -
      ( min_neg.at(i) * min_dist.at(i)      * min_dist.at(i)      * min_dist_pos.at(i) ) -
      ( minimi.at(i)  * min_dist_neg.at(i) * min_dist_pos.at(i) * min_dist_pos.at(i) ) );


  db = (  ( minimi.at(i)   * min_dist_pos.at(i) * min_dist_pos.at(i) ) +
      ( min_neg.at(i)  * min_dist.at(i)      * min_dist.at(i) )      +
```

```
      ( min_pos.at(i)   * min_dist_neg.at(i) * min_dist_neg.at(i) ) -
      ( minimi.at(i)    * min_dist_neg.at(i) * min_dist_neg.at(i) ) -
      ( min_pos.at(i)   * min_dist.at(i)     * min_dist.at(i) )     -
      ( min_neg.at(i)   * min_dist_pos.at(i) * min_dist_pos.at(i) )   );


  dc = (  ( min_pos.at(i) * min_dist.at(i) )      +
      ( minimi.at(i)    * min_dist_neg.at(i) ) +
      ( min_neg.at(i)   * min_dist_pos.at(i) ) -
      ( min_neg.at(i)   * min_dist.at(i) )      -
      ( minimi.at(i)    * min_dist_pos.at(i) ) -
      ( min_pos.at(i) * min_dist_neg.at(i) )      );


    a = da / delta;
    b = db / delta;
    c = dc / delta;
    vortex = - b / ( 2 * c );
    vortey = - ( b * b - 4 * a * c ) / ( 4 * c );
    xverticiMIN.push_back(vortex * 0.05);
    yverticiMIN.push_back(vortey);
  }


  vector<int> eliminamassimi; //vector contenente la posizione dei valori
    da eliminare
  vector<int> eliminaminimi;
  int temperasemax;
  int temperasemin;



  for (int i = 0 ; i < max_size -1 ; i++) //Permette di trovare eventuali
    massimi fasulli
  {
    if (abs (xverticiMAX.at(i) - xverticiMAX.at(i+1)) < 0.8 ) //Intervallo
    considerato errore
    {
      if ( yverticiMAX.at(i)  -  yverticiMAX.at(i+1)  >= 0)
      {
        temperasemax = i+1;
        eliminamassimi.push_back(temperasemax);
      } else if ( yverticiMAX.at(i) -  yverticiMAX.at(i+1)  < 0 &&
    temperasemax != i)
        {
        temperasemax = i;
        eliminamassimi.push_back(temperasemax);
        }
    }
  }


  for (int i = 0 ; i < min_size - 1; i++) //Permette di trovare eventuali
```

```
    minimi fasulli
  {
    if (abs (xverticiMIN.at(i) - xverticiMIN.at(i+1)) < 0.8 )
    {
      if ( yverticiMIN.at(i) - yverticiMIN.at(i+1) <= 0 && temperasemin != i )
      {
        temperasemin = i+1;
        eliminaminimi.push_back(temperasemin);
      } else if ( yverticiMIN.at(i) - yverticiMIN.at(i+1) > 0 )
       {
        temperasemin = i;
        eliminaminimi.push_back(temperasemin);
       }
    }
  }


  int puliziamax = eliminamassimi.size();
  int puliziamin = eliminaminimi.size();
  int f = 0; //Variabile necessaria per regolare la modifica posizione vettori

  for (int i = 0 ; i < puliziamax ; i++) //Pulisce il vector definitivo di massimi
  {
    xverticiMAX.erase( xverticiMAX.begin() + eliminamassimi.at(i) - f );
    yverticiMAX.erase( yverticiMAX.begin() + eliminamassimi.at(i) - f );
    f++;
  }

  f = 0;

  for (int i = 0 ; i < puliziamin ; i++) //Pulisce il vector definitivo di minimi
  {
    xverticiMIN.erase( xverticiMIN.begin() + eliminaminimi.at(i) - f );
    yverticiMIN.erase( yverticiMIN.begin() + eliminaminimi.at(i) - f );
    f++;
  }

  double max_v_size  = xverticiMAX.size();
  double min_v_size = xverticiMIN.size();


    //cout << "Le x dei massimi valgono: " << endl;
  for (int i = 0 ; i < max_v_size ; i++)
  {
    cout << xverticiMAX.at(i) << "\t" << yverticiMAX.at(i) << endl;
  }


  //cout << "Le x dei minimi valgono: " << endl;
```

```
  for (int i = 0 ; i < min_v_size ; i++)
  {
    cout << xverticiMIN.at(i) << "\t" << yverticiMIN.at(i) << endl;
  }


/*

   int q = 0;                    //Algoritmo per la visualizzazione dei risultati
  cout << "Massimi: " << endl;
  for (double massimi)
  {
    cout << massimi.at(q) << endl;
    q++;
  }
  q = 0;
  cout << "A una posizione di: " << endl;
  while (max_dist.at(q) != 0)
  {
    cout << max_dist.at(q) << endl;
    q++;
  }


  q = 0;
  cout << "Minimi: " << endl;
  while (minimi.at(q) != 0)
  {
    cout << minimi.at(q) << endl;
    q++;
  }
  q = 0;
  cout << "A una posizione di: " << endl;
  while (min_dist.at(q) != 0)
  {
    cout << min_dist.at(q) << endl;
    q++;
  }
*/
  return 0;

}
```

../src/massimi_minimi.cpp